elp build embedded systems

Gary Legg, Senior Technical Editor

Thanks largely to easy-to-use design tools, fuzzy logic is now gaining a foothold in embedded systems. Although large numbers of commercial fuzzy-logic applications have yet to materialize (especially in the US), numerous successful test projects indicate that many fuzzy products will soon follow. For the most part, product developers will implement these products in surprisingly compact code in conventional microcontrollers (μCs).

Fuzzy logic is slowly catching on, and fuzzy design tools are getting better: They don't just help design fuzzy logic, they let you create complete embedded systems.

Real commercial applications, however, require that fuzzy tools do more than just aid the design process. Because fuzzy code almost always coexists with conventional code in embedded systems, fuzzy tools must help combine the two. Increasingly, the tools are doing just that.

Better system capabilities

The latest fuzzy-logic tools provide substantial improvements in system-development capabilities. Intel's Fuzzybuilder, for example, merges fuzzy tools with Apbuilder, a software tool that automatically generates peripheral-initialization code for MCU-96 μCs. Also included is Model-builder, a software package that lets you create a system-level application model without writing any code. The fuzzy part of Fuzzybuilder comes from Inform, which also supplies a remote real-time debugger for fuzzy applications. Similarly, Togai InfraLogic includes remote debugging in its tools' capabilities.

Fuzzy tools are also linking with other tools: some fuzzy, some not. For example, Fuzzy Systems Engineering's Fuzzy Knowledge Builder (previously Manifold Graphics Editor) now creates input to ByteCraft's Fuzz-C Preprocessor, thus linking a front-end fuzzy design tool to a C-like language that appeals to designers already versed in conventional system development. For Windows-based applications, HyperLogic's CubiCalc and some other fuzzy tools now link fuzzy programs to other programs via dynamic data exchange (DDE).

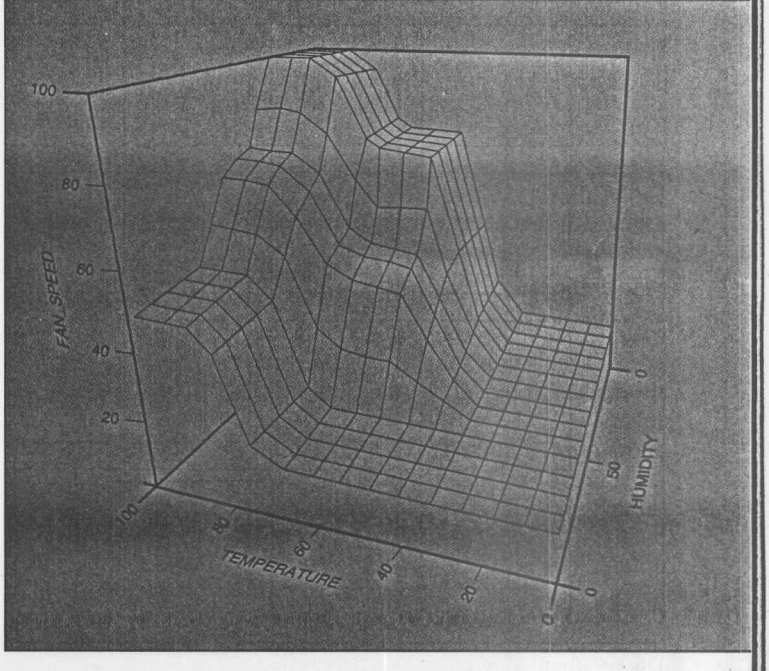

Fig 1—Simulators in fuzzy-logic tools can draw control-surface plots to show system response. Byte Dynamics' Fuzzy Logic Designer drew this plot.

Fuzzy-logic tools

Because the adaptation of fuzzy logic has been slower than expected, some fuzzy-tool manufacturers are scrambling to survive. As a result, choosing the right fuzzy tools to use—difficult enough in itself—can be complicated by concerns over whether or not a tool provider will remain viable.

Companies suffering most from the sluggish pace of fuzzy logic's commercial proliferation are those that have concentrated solely on fuzzy-logic products. Togai InfraLogic, Aptronix, and NeuraLogix, for example, have all put a great deal of effort into fuzzy products that have been slow to take off. As a result, Togai has scaled back, Aptronix has branched into fuzzy-logic consulting, and NeuraLogix has relocated from Florida to California's Silicon Valley. Of all the companies concentrating exclusively on fuzzy-logic tools, only pioneer HyperLogic has remained healthy. Other companies with fuzzy tools either augment their income with consulting and training in fuzzy logic or have developed fuzzy tools as an outgrowth of regular business.

Semiconductor partnerships

Semiconductor companies with a stake in fuzzy logic—notably companies that manufacture μCs—have formed relationships with third-party tool vendors. Inform, a German company that has offices in the US, is the most prominent of the third parties, having established partnerships with Intel, SGS-Thomson, Siemens, Texas Instruments (for fuzzy DSP applications), and Microchip Technologies. Chip-maker Hitachi has a partnership with Togai, as does VLSI Technology. Motorola works with Togai, Aptronix, and Hiware and also has in-house tool development. Of all the major chip makers, only National Semiconductor has developed its fuzzy tools solo.

Europe and Japan are well ahead of the US in fuzzy applications, and Europe, not surprisingly, is a major source of fuzzy design tools. Inform i the best known of the fuzzy compa nies, but Switzerland's Hiware is no getting attention from its allianc with Motorola. Another Swiss compa ny, Fuzzysoft, also develops fuzz tools; it markets through GTS Traut zl GmbH of Germany. Togethe European and American fuzzy-to vendors add up to a sizable numbe (**Table 1**).

Even some enthusiastic developers fuzzy applications have been slow t adopt fuzzy tools, however. You can g by without special tools, they repor because fuzzy-logic software isn't a that complicated (**Ref 1**). Typical fuz controllers often run in only a few hu dred bytes of code.

But to ignore the benefits of fuz tools, say other developers, is to igno the benefits of fuzzy logic itself. Fa prototyping is one of fuzzy logic's ma strengths, according to fuzzy-desig consultant David Brubaker, and prot typing is where fuzzy tools work be Tools that can simulate a fuzzy desig performance are especially useful, sa Brubaker. The simulator in HyperLo ic's CubiCalc often assists Brubaker quick feasibility studies, which tu into actual design projects.

Multiple capabilities

Various fuzzy design tools provi one or more of several capabilities:

- Help to design a fuzzy control sy tem with CASE (computer-aid software-engineering) capabiliti
- Simulate a fuzzy design to help v idate it, as previously mentioned
- Generate code, in C or in vario assembly languages, for impleme ing a fuzzy design
- Help to merge the fuzzy code wi other code
- Help debug the entire system.

Using fuzzy tools to design a fuz controller requires knowing how fuz logic works (**Refs 2** to **5**). Essential says Brubaker, you have to learn how work in a different domain. Fortuna ly, that domain offers advantages th make your effort to learn worthwhi

Different tools support the vario fuzzy design steps in a range of ways and to varying degrees. At a minimu however, all tools let you do thr things: define fuzzy input and outp

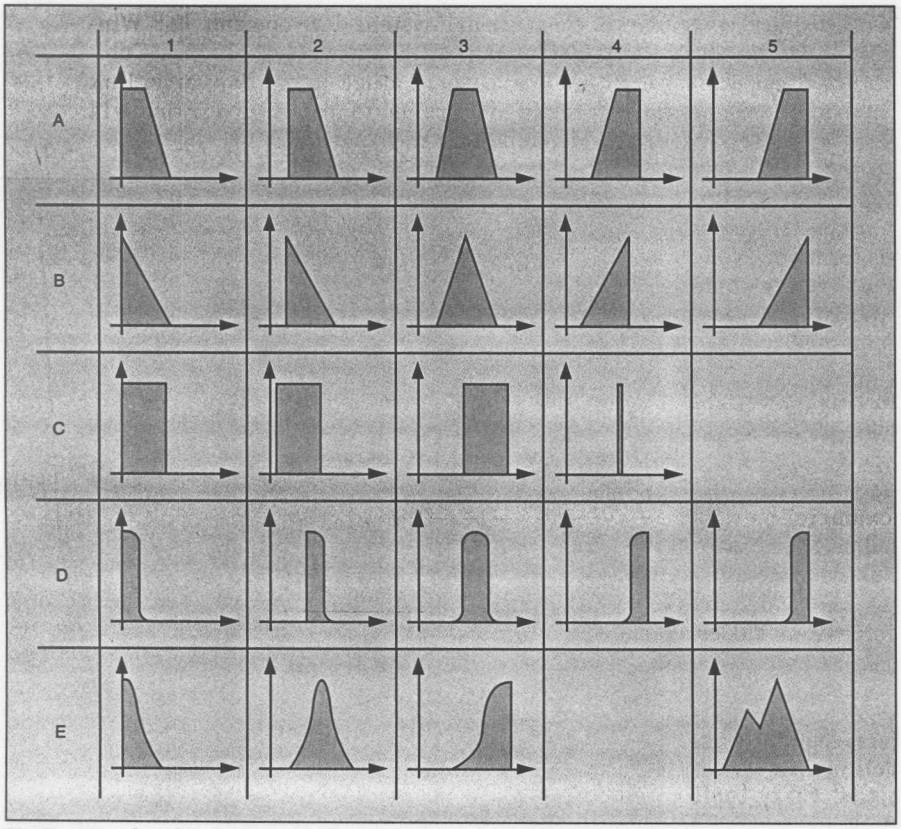

Fig 2—Membership functions take a variety of shapes, as this selection screen from GTS Trautzl's FS-Fuzzysoft shows. The most commonly used functions are triangular and trapezoidal.

iables; specify fuzzy sets (also called mbership functions or adjectives) it correlate fuzzy variables with zy values; and define linguistic rules it describe the desired system iavior.

Once you have a fuzzy design, most of tools automatically generate code implementing it. Usually, the code is C or assembly language for one of a mber of μCs. A few tools do provide ier languages, however (see **box**, eatures to look for"), including Basic, rtran, and Ada. Often, fuzzy-tool-nerated source code contains clear planatory comments about the fuzzy erations it implements.

ining with a simulator

A simulator helps you "tune" your zzy design. Tuning, an iterative ocess of refining a fuzzy design, is ually the most time-consuming part creating a fuzzy system. Without nulation, you can get into a lengthy p of design modifications, code gen-ation, downloading, and debugging. ith simulation, you see the results of sign changes almost immediately.

Simulators provide displays that monstrate system behavior. These splays include both 3D plots of control rfaces (also called decision surfaces) d 2D orthogonal slices through the ntrol surfaces. The 3D plots usually ow one output variable as a function two input variables (**Fig 1**); alterna-vely, they may show an error term as function of inputs. Many simulators t you alter plot perspective by chang-g your apparent vantage point. To lp compare the results of changes on fferent output variables, some simu-tors also let you view multiple 3D ndows simultaneously.

With some tools, you can make sign changes on the fly in the tuning ocess and see their simulated effects mediately. Typical modifications clude changing the shapes of fuzzy embership functions, changing the lues of variables, and defining new ts. You can usually change rules, too, hough some restrictions may apply. r example, with some tools, you may ve to restart the simulator when you ange rules. With Togai's simulator, u can always delete and disable rules the fly, but you'll have to make provisions when you set up your system if you want the option to add rules.

Remote, real-time debuggers help wring out a fuzzy controller after it's running in a target system. Like simulators, real-time debuggers let you alter, add, or delete membership functions and rules. Unlike simulators, these debuggers provide feedback from the embedded system itself. A communications link exists, usually via an RS-232C port, between the host and target systems. Remote debuggers are available from Inform and Togai.

Features for fuzzy design

At the heart of most fuzzy tools is the capability for creating the design for a fuzzy control system. With features such as graphics-based function editors and linguistic fuzzy-rule editors, the tools let you work intuitively in the fuzzy domain, at a level of abstraction above that of ordinary programming.

You begin a fuzzy design by specifying input and output variables and by defining membership functions. Membership functions typically have names like "warm" (to describe, for example, a measured temperature) or "fast." They express the fuzzy designer's intuitive concept of which temperatures correspond to "warm" and which speeds are fast. Depending on the fuzzy tool, you can create membership functions with a graphics editor, a text editor, or both.

Membership functions can have many different shapes (**Fig 2**), although not all fuzzy tools support all shapes. The most common shapes are triangles (pyramids) and trapezoids (truncated pyramids), which are usually symmetrical, but don't have to be. Membership functions can have many other shapes and even can be nonlinear, although nonlinear functions usually get converted to piecewise linear approximations when implemented in μCs. Output membership functions (sometimes called consequence functions) can also include singletons, which denote fixed values.

Simple membership functions, such as those with triangular and trapezoidal shapes, are adequate for most applications. In most cases, the smoothness of system response depends more on function overlap and defuzzification technique than on function shape. Really complex applications may require more complex shapes and, most likely, additional computing power—a 16-bit controller instead of an 8-bit device, for example. Nonlinear membership functions are usually confined to workstations, says Brubaker. For μC implementations, storage and computational costs are prohibitive.

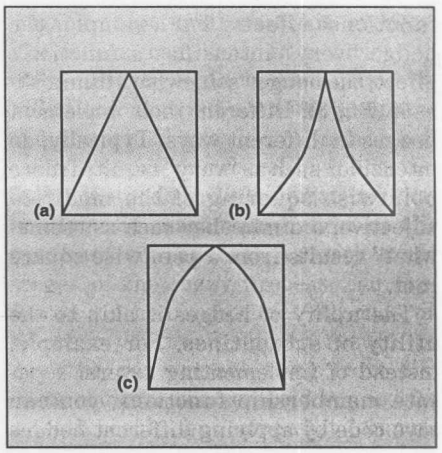

Fig 3—Hedges modify membership functions. Modifying an original triangular function (a) by the hedge "very" yields (b); modification by the hedge "somewhat" yields (c).

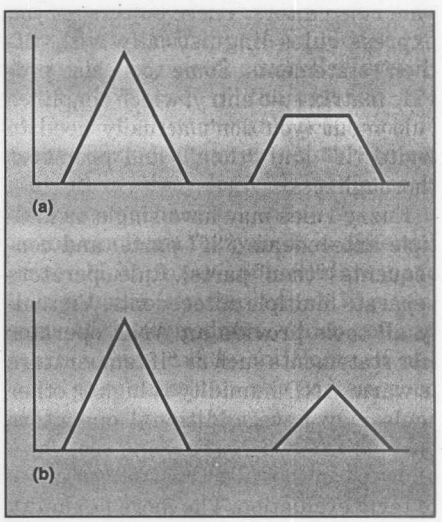

Fig 4—Applying a rule-activation level to output membership functions results in two types of scaling: by minimum (a), which truncates the function, and by multiplication (b), which retains its general shape.

Functions on functions

Some fuzzy design tools let you modify membership functions with "hedges" to accentuate or diminish the

functions' effects. For example, the hedge "very" intensifies a function's effect; the hedge "somewhat" diminishes it (**Fig 3**). Different tools implement hedges in different ways. Typically, an intensifier such as "very" results from a pointwise squaring of the modified adjective; a diminisher such as "somewhat" results from a pointwise square root.

The utility of hedges is akin to the utility of subroutines. For example, instead of implementing several separate membership functions, you can save code by applying different hedges to a single function. Also, like subroutines, hedges aid conceptual understanding. Essentially, they apply a function to a function.

To help you define the rules for controlling a fuzzy system, fuzzy tools provide rule editors. These editors let you express rules linguistically with "if-then" statements. Some tools also provide matrix rule entry, which simplifies rule input (you don't actually need to write "if" and "then") and promotes thoroughness.

Fuzzy rules may have single or multiple antecedents ("if" parts) and consequents ("then" parts). Rule operators separate multiple antecedents. Virtually all tools provide an AND operator (for statements such as "if temperature is warm AND humidity is high"); other tools may have additional operators such as OR and NOT. Some tools also allow parentheses to control the order of term evaluation. The more flexibility a tool offers in writing rules, the more you can express rules in the way you naturally think. (Do not assume, however, that having fewer rules with more terms is better. Sometimes, reducing the number of rules is more trouble than it's worth.)

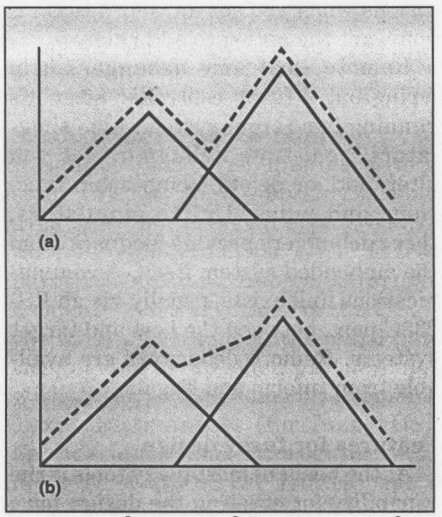

Fig 5—Combining a rule's output-membership functions shows the rule's overall effect. The combined result can be the maximum points of the two functions (a) or the sum (b).

What the fuzzy code does

The code a fuzzy tool generates must perform all the steps required of a fuzzy-logic controller. These steps include fuzzification (to convert input "crisp" values to fuzzy values), inferencing (application of fuzzy rules), and defuzzification (to determine crisp output values). Different tools perform the steps in different ways, and some individual tools provide choices of methods for some steps.

Fuzzification determines membership degrees (from 0 to 1) of input variables in fuzzy sets, which are denoted by input membership functions (partial membership is permissible in fuzzy set theory). For a simple single-antecedent rule, the degree of membership determines the degree to which the rule is active or the degree to which the rule's specified output actions occur. The methods the fuzzy tools provide for determining degree of membership range from simple table look-up to interpolation to computation of nonlinear functions. Obviously, the different methods make different demands on a fuzzy system for memory and computational power.

For multiple-antecedent rules, fuzzy inferencing combines the effects of all antecedents to determine a rule's overall effect. For two antecedents separated by an AND operator, the result most fuzzy tools supply is the minimum of the antecedents' degrees of membership in their respective fuzzy sets. For antecedents separated by an OR operator, the result is usually the maximum of the two numbers. Some tools (FS-Fuzzysoft, for example) provide more than simple minimum and maximum choices, but using this type of tool is probably beyond a novice designer's capabilities.

A rule's activation level, determined by combining antecedents' effects, applies to the rule's output membership function(s) to determine how much the rule contributes to system behavior. (In simple terms, the degree of truth of a rule's "if" part determines the degree

LOOKING AHEAD

One advantage fuzzy logic has over conventional control systems is its flexibility in accommodating fairly broad variations in system parameters. In the future, however, fuzzy logic will be even more flexible. The components of a fuzzy system will be variable—rather than fixed—so the system itself can change as conditions change.

In adaptive fuzzy-logic systems, rules and fuzzy sets (membership functions) can change on the fly. Some fuzzy design tools may add support for adaptive systems as early as this year, although many fuzzy designers may not be ready yet for that next level of abstraction.

Although many designers still are learning how to use fuzzy logic's most basic capabilities, the long-range trend will be toward creating more sophisticated systems. For example, fuzzy-logic consultant David Brubaker reports already having designed a system with two fuzzy controllers—one that actually provides system control and another that adapts the first one to changing conditions. The combination of fuzzy control and conventional proportional-integral-differential (PID) control will also be more prevalent. An adaptive system could even use a fuzzy controller to adapt a PID controller to changing conditions. Eventually, genetic algorithms will help fuzzy systems adapt.

In the meantime, the use of neural networks with fuzzy systems will grow. And that growth will be assisted by more and better fuzzy design tools with neural-net modules.

of application of the rule's specified output actions.)

Application of the rule activation level to the output membership function most often occurs in one of two ways (**Fig 4**): clipping (sometimes called scaling via minimum) or scaling (also called scaling via multiplication). If your fuzzy tool provides a choice, determining which scheme to use can depend on your having a good working knowledge of fuzzy logic. However, some tools that offer both schemes help you make a choice according to the type of application you're designing. HyperLogic's CubiCalc, for example, defaults to the multiplicative scaling option if you've specified that the system you're designing should have a smooth response. Alternatively, the purpose of some fuzzy systems is to make distinct choices, in which case the response should not necessarily be smooth.

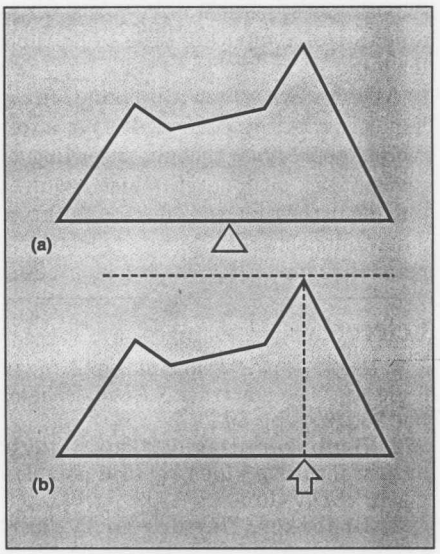

Fig 6—Defuzzification converts combined output-membership functions to a single real value. Two common methods select output values from a centroid (a) and the point of maximum height (b).

Schemes for combining rules

To determine the combined effects of all rules, fuzzy inferencing combines output fuzzy sets (consequents) to create a resultant fuzzy set. Again, various combination schemes exist; CubiCalc's options (**Fig 5**) illustrate the general idea. In **Fig 5**, the combination of two sets may result in the maximum point-by-point values of the two sets, the added pointwise values, or simply the "best" of the two sets. In general, "max" and "sum" tend to be better for continuous response, and "best" is better for distinct choices. The CubiCalc user's manual offers more detail.

Ultimately, the process of defuzzification computes a "crisp," numerical output value from scaled and combined membership functions. This process, like others, can occur in a variety of ways (**Fig 6**). One method, usually called the centroid method, computes the center of gravity of combined, scaled-output membership functions. Other methods include variations that involve the maximum height of the resultant consequent. As with other stages of the fuzzy process, the preferred method sometimes depends on whether you need smooth results or whether you are selecting from discrete choices. The centroid is good for smooth results but somewhat compute intensive.

Knowing which fuzzy options to use, or even which ones to require in a fuzzy design tool that you buy, depends on a level of knowledge too great for an article of this length to convey. However, it's often true that fuzzy logic's robustness enables you to create a very good controller even if you haven't made all the best possible choices. Most important, according to many fuzzy designers, is simply to get started in fuzzy logic. Experience is the best teacher.

You can learn fuzzy logic in different ways. Most fuzzy-tool vendors have inexpensive, limited-capability versions of their products, which you can learn to use hands-on. Some of the tools' user's manuals also contain good

FEATURES TO LOOK FOR

Unless you're ahead of most others on the learning curve, you probably won't need fuzzy-logic design tools with the most exotic features. Your primary concern, in fact, might simply be what processors the generated fuzzy code will run on.

Most fuzzy tools are available in versions for generating assembly-language code for numerous µCs. Other tools produce code for Texas Instruments TMS320 DSPs and for fuzzy processors and coprocessors from VLSI Technology and NeuraLogix. Most tools also generate C code.

In addition to C, some tools produce code in other high-level languages. Fuzzle, from Modico, generates Fortran; Fuzzy Knowledge Builder, from Fuzzy Systems Engineering, generates both Fortran and Basic. RT/Fuzzy Module, part of a larger system-development package from Integrated Systems, produces C and Ada code. The Fuzz-C Preprocessor from ByteCraft lets you write in a C-like language created just for fuzzy systems.

A simulator is an especially good feature to have in a fuzzy tool. Simulators are available in many tools; some others can link to separate simulators, such as MatLab.

A matrix rule editor is also nice. Implemented first by Fuzzy Systems Engineering, this feature is now in several other tools. A matrix editor simplifies rule entry and promotes thoroughness by displaying your entire rule base in tabular form.

A graphical membership-function editor can be helpful, although sometimes a text editor is more convenient. Try to select a tool with both—and then take your pick.

Membership functions more complex than triangles and trapezoids may go unused until you're a fuzzy expert, but they do reflect a tool's sophistication. Similarly, hedges are for the well initiated.

A remote real-time debugger will appeal to engineers with embedded-system experience. Unlike simulators, real-time debuggers let you know what's happening in the processor that runs your fuzzy code.

The control-systems specialist may benefit from FS-Fuzzysoft (GTS Trautzl), which allows mixing of classical and fuzzy techniques.

Fuzzy-logic tools

tutorial information, and many fuzzy tool companies offer training in fuzzy-logic design.

Automation via neural nets

You may be able to bypass some learning by using a tool with neural-network capabilities. The neural nets in these products can "learn" how a system should work by processing large numbers of input and output data sets. With that knowledge, the tools then generate the code for a fuzzy system with much less user involvement than would otherwise be required. Neural-net tools for fuzzy applications are available from Inform, Motorola, National Semiconductor, and Togai. National recently updated its offering to produce C code; the earlier version generated only assembly code for COP8 controllers.

If neural nets spare you the effort of learning fuzzy logic, however, they may exact a price in demanding at least a rudimentary knowledge of neural networks. They can also require large amounts of computing time to do their learning.

As with every new discipline, fuzzy logic has a learning curve. It's not a terribly steep curve, though, according to many fuzzy designers; it just requires putting in the time and effort to shift into a new mindset. Most designers who have made the effort say it's worth it. EDN

References

1. Legg, Gary, "Transmission's fuzzy logic keeps you on track," *EDN*, December 23, 1993, pg 60.
2. Brubaker, David, "Fuzzy-logic basics: intuitive rules replace complex math," *EDN*, June 18, 1992, pg 111.
3. Brubaker, David, "Fuzzy-logic system solves control problem," *EDN*, June 18, 1992, pg 121.
4. Legg, Gary, "Special tools and chips make fuzzy logic simple," *EDN*, July 6, 1992, pg 68.
5. Conner, Doug, "Designing a fuzzy-logic control system," *EDN*, March 31, 1993, pg 76.

Acknowledgments

Dr David Brubaker of Huntington Advanced Technology (Menlo Park, CA, (415) 325-7554) provided useful comments, suggestions, and information for this article. David's fuzzy-logic column premieres in this issue of EDN and will appear every other issue.

Special thanks to HyperLogic (Escondido, CA, (619) 746-2765) for illustrations and concept explanations from the CubiCalc user's manual. The CubiCalc manual is a good source of information about fuzzy-logic design.

Contact Senior Technical Editor Gary Legg at (617) 558-4404; fax (617) 558-4470.

Text continued on pg 58

VOTE

Please use the Information Retrieval Service card to rate this article (circle one):

High Interest	Medium Interest	Low Interest
589	590	591